IN THE SPECIFICATION:

Please amend the specification as follows:

Please amend the paragraph on page 11, from lines 18 through 20, as follows:

Fig. 30 illustrates a flowchart of a disk-array write program [[2403]] of the embodiment 3 according to the present invention.

Please amend the paragraph on page 15, from lines 14 through 25, as follows:

The disk array controller 122 can permit the host 1 to treat a plurality of physical drives as a single logical drive. For example, the disk array controller 122 permits the host 1 to treat two disk drives 117, 118 as a single logical drive 213, and in addition to it, to treat three disk drives 119, 120, 121 as a single logical drive 214. To be more specific, it looks as if not disk drives 117 through 121 as physical drives but the logical drives 213, 214 are connected to the host 151. In the following description, it is assumed that drives simply mean not physical drives ([[215]] 117 through [[219]] 121) but logical drives (213, 214).

Please amend the paragraph bridging pages 17 and 18, from line 24 on page 17 through line 19 on page 18, as follows:

Fig. 5 is a diagram illustrating a disk area management method of a file system of the OS 2. The file system of the OS 2 divides a drive area into a plurality of fixed-length blocks 501 for management. When a data length of a file is longer than the fixed-length block 501, the file system divides the file into a plurality of blocks and stores the blocks. In this case, these blocks may not always be allocated to consecutive areas. If a data length of a file is longer than the fixed-length block 501 [[401]], the file system divides the file into a plurality of blocks and stores the blocks. In this case, these blocks may not always be allocated to consecutive areas. Therefore, the file system is required to manage locations of the disk areas where the file is allocated. File management information of the OS 1 [[2]] is different from that of the OS 2. Moreover, this embodiment is based on the assumption that a block length of the fixed-length block 401 [[501]] is four times as long as that of the fixed-length block 501 [[401]]. It is to be noted that each of two areas, which are illustrated by bold lines, indicates a partition area.

Please amend the paragraph bridging pages 20 and 21, from line 19 on page 20 through line 8 on page 21, as follows:

Reference numeral 810 shows contents of the file management information 703 (or 706) of the OS 2. Reference numeral 811 is an area in which a file name is stored. In this embodiment, a file name "file A" is stored. ~~In this embodiment, a file name "file A" is stored.~~ Following the area 811, file management information about the file A is stored. The file management information of the file A comprises: a file attribute area 812 for storing information about file access permitted/not permitted; a time stamp area 813 for storing file update date; a file size area 814 for storing a file size; a pointer area 815 indicating an area in which data of the file A is stored; and pointers 816 through 824 that point to disk areas in which the file A is stored.

Please amend the paragraph bridging pages 21 and 22, from line 9 on page 21 through line 8 on page 22, as follows:

Reference numerals 831 and 832 are data blocks, which are pointed to by the pointers 808 and 809 respectively. Block lengths of these data blocks are the same as the block length of the block 401. In addition, reference numerals 833 through 840 are data blocks, which are pointed to by the pointers 816 through [[825]] <u>824</u> respectively. Block lengths of these data blocks are the same as the block length of the block 501. Reference numeral 830 is a whole data of the file A. As described above, the block length of the block 401, which is a unit of management by which the file system of the OS 1 manages disk areas, is four times as long as the block length of the block 501 of the OS 2. Therefore, the number of pointers used by the OS 2 for managing data of the file A is four times as many as that used by the OS 1. For example, if the file A is created by an application program on the OS 1, a block length, that is to say, a unit of management for data areas must be changed in order to permit the OS 2 to recognize the file A. In order to permit the OS 2 to recognize the file A that has been created by an application program on the OS 1, a processing, which converts the contents 800 of the file management information of the OS 1 into the contents 810 of the file management information of the OS 2, is required. The processing will be described later.

Please amend the paragraph on pages 22 through 24, from line 9 on page 22 through line 1 on page 24, as follows:

Fig. 9 is a diagram illustrating a configuration of a host table <u>110</u>. A host ID is an ID of each host that accesses a disk array controller. In this embodiment, as shown in Fig. 1, because the host 1 and the host 2 are connected to the disk array controller 121, host IDs 01, 02 are assigned to the hosts respectively. When a host issues an input/output request to a disk

array controller, a host ID is an identifier that is transmitted from the host to the disk array controller, and that is transferred together with the input/output request. In "type of file system", a type of a file system operating in each host is stored. In "type of target file system", a type of a file system, which creates a file stored in a storage system, is stored. As regards the host 1 (host ID 01), file systems, which are described in the column of "type of file system" and in the column of "type of target file system" are the same. This means that a file system for creating a file is the same as a file system by which input/output processing is desired. Therefore, the host 1 can access data in the storage system. On the other hand, as regards the host 2 (host ID 02), a file system described in the column of "type of file system" is different from that described in the column of "type of target file system". This means that a file system for creating a file is different from a file system by which input/output processing is desired. In the prior art, the OS 2 cannot access data in the storage system. A unit of management [[1104]] indicates a unit of management of a file system. For example, a unit of management of the OS 1 file system, that is to say, a data length of the block 401 is 4096 bytes; and a unit of management of the OS 2 file system, that is, a data length of the block 501 is 1024 bytes. There are several methods for setting the host ID, the type of file system, the type of target file system, <u>and</u> the unit of management. For example, although not illustrated, there are the following methods: a method in which a user sets them from a console of the disk array controller; a method in which a special command is issued to the disk array controller from the host to transfer information about the host and to set them; a method in which the disk array controller judges from access procedures of the file system automatically to set them; and the like.

Please amend the paragraphs on page 24, from lines 6 through 24, as follows:

Fig. 11 shows a flowchart of the partition conversion program [[1202]] <u>1001</u>. The partition conversion program [[1202]] <u>1001</u> converts disk management structure of the OS 1 into disk management structure of the OS 2. The partition conversion program [[1202]] <u>1001</u> is a program used for permitting the file system of the OS 2 to recognize a partition that exists in the file system of the OS 1.

For example, the partition conversion program [[1202]] <u>1001</u> is started in the following cases: when power of a disk device is turned on; at the time of a user's instruction; in a case where there is an access from a host of a different kind (a host on which a different file system operates) for the first time; or the like. As regards judgment whether or not an input/output request is issued from a host of a different kind, the "type of target file system"

and the "type of file system" in the host table in Fig. 10 are compared. If they are not the same, the input/output request is judged to be an access from a host of a different kind.

Please amend the paragraph on pages 24 through 26, from line 25 on page 24 through line 1 on page 26, as follows:

In a step 1101, partition information of an original file system is read. In this embodiment, partition information of the file system of the OS 1 is read. In a step 1102, a block number of a first block of a partition is converted. A unit of disk management of the OS 1 is different from that of the OS 2. Therefore, when a top of a partition is described using a block number, the same disk address corresponds to a different block number. This is the reason why the block number is converted. In a step 1103, a partition size is converted. It is also the processing performed for the same reason as the step [[1302]] 1102. When a partition size is described using the number of blocks, and when units of management are different, the same partition capacity is not equivalent to the same number of blocks. This is the reason why the partition size is converted. In a step [[1104]] 1103, whether or not all partitions have been converted is judged. If there is a partition that has not been converted yet, a process returns to the step [[1101]] 1102. A result of the partition conversion (hereinafter referred to as "partition information after conversion") is stored in, for example, the cache area 108 in the disk array controller 122 as shown in Fig. 13. If the partition information after conversion 1301 is accessed, it is possible to recognize a partition 1 (1307) and a partition 2 (1308) in the file system of the OS 1 even from a different file system.

Please amend the paragraph on pages 26 through 28, from line 15 on page 26 through line 13 on page 28, as follows:

The file system conversion program 1002 will be described while dividing its operation into steps. In a step 1201, file management information of an original file system is read. In this embodiment, file management information of the file system of the OS 1 is read. In a step 1202, a block number of a first block of a file is converted. This means that information of the reference numeral 802 in Fig. 8 is converted into information of the reference numeral 815 in Fig. 8. This is based on the same reason as that of the step 1102 described in Fig. 11. In a step 1203, an owner of the file is converted. In this case, if use of a specific owner as a file owner after conversion is desired because of file protection, or the like, an owner after conversion is instructed beforehand. In addition, depending on a format of a file system, there is a case where no owner exists. In such a case, this step is omitted. In

a step 1204, a time stamp of a file is converted. This means that data of the reference numeral 805 in Fig. 8 is converted into a data format of the reference numeral 813 in Fig. 8. In a step [[1505]] 1205, a file attribute is converted. This shows that data of the reference numeral 804 in Fig. 8 is converted into a data format of the reference numeral 812 in Fig. 8. In this case, writing a write-protect attribute to a file attribute after conversion can configure file systems as follows: an original file system can update the file whereas a different file system cannot update it. In a step 1206, a block number of a block, in which data of a file is stored, is converted. This shows that it is converted into a data format of the reference numerals 816, 817, ... shown in Fig. 8. In this case, a pointer pointing to a data area is converted so as to point to a data area that is stored by the file system of the OS 1. This permits a plurality of hosts to refer to the same data. In this embodiment, a unit of disk management of the OS 1 is four times as large as that of the OS 2. Because of it, by processing of a step 1206, data, which is pointed to by the pointers 808 and 809, is converted into data stored in eight areas, which are pointed to by the pointers 816 through 819 and the pointers 821 through 824. In a step 1207, an end of a file is judged. If conversion of one file is not completed, the step 1206 is repeated. In a step 1208, whether all file information in a partition has been converted or not is judged. In a step 1209, whether all file information in all partitions in a drive has been converted or not is judged. As a result of the processing described above, the conversion of the file information to the different file system is completed.

Please amend the paragraph bridging pages 30 through 31, from line 10 on page 30 through line 22 on page 31, as follows:

Fig. 16 illustrates a flow of the disk-array read program 106. In a step 1601, a host ID of a host, which has issued an access request, is obtained. In a step [[1802]] 1602, a "type of file system", a "type of target file system", and a "unit of management", which correspond to the obtained host ID, are obtained from the host table 110. In a step 1603, whether or not it is an original file system of the host, which has issued an access request, is judged. If it is the original file system, a process proceeds to a step [[1810]] 1610. If not, the process proceeds to a step [[1804]] 1604. In this embodiment, if it is an access from the host 1, the process [[will]] proceeds to "YES"; or if it is an access from the host 2, the process [[will]] proceeds to "NO". Steps 1604 through 1609 are executed when there is a request from a file system, which is different from the file system that has stored the file in the disk device. In a step 1604, whether or not it is an access to a boot program is judged. If it is the access to the boot

program, the process proceeds to a step 1607. If not, the process proceeds to a step 1605. In the step 1607, a boot program corresponding to the file system after conversion is read. In the step 1605, whether or not it is an access to partition information is judged. If it is the access to the partition information, the process proceeds to a step 1608. If not, the process proceeds to a step 1606. In the step 1608, the partition information after conversion is read. In a step 1606, whether or not it is an access to file management information is judged. If it is the access to the file management information, the process proceeds to a step 1609. If not, the process proceeds to a step 1610. In the step 1609, the file management information after conversion is read. In the step 1610, usual disk-array read processing is performed, and data corresponding to a requested disk address is read. In a step 1611, read data are transferred to the host. By executing these steps, a file read request from a plurality of hosts, on which different OSs operate, can be processed.

Please amend the paragraphs on pages 33 through 34, from line 9 on page 33 through line 17 on page 34, as follows:

Fig. 18 illustrates a flow of a disk-array read program of the embodiment 2. As regards steps that are common to those in the flow described in Fig. 16, description will be omitted. Only steps, which are different from those in the flow in Fig. 16, will be described. In the flow described in Fig. 18, a step [[1901]] 1801 is added. To be more specific, if a file system of a host, which has issued an access request, is an original file system, an offset 1703 (refer to Fig. 17) is added to a disk address. This is because the OS 2 boot program 1501, the partition information after conversion 1301, and the file management information after conversion 1401, 1402 are stored in the disk drives. Adding an offset to a disk address, to which a host requests an access, enables an access to correct data.

It is to be noted that even when the OS 2 boot program 1501, the partition information after conversion 1301, and the file management information after conversion 1401, 1402 are stored in the disk drives, frequently accessed data is loaded [[load]] into the cache area 108, which enables ~~us to acquire~~ acquiring the same read performance as that in the embodiment 1. Moreover, in the embodiment 2, although the OS 2 boot program 1501, the partition information after conversion 1301, and the file management information after conversion 1401, 1402 are allocated to specific areas in the disk drives, the allocation of the data can be changed freely. Therefore, even if an OS of a host connected to a storage system is changed, or a new host is additionally connected, allocating the boot program, the partition information after conversion, and the file management information after conversion to space areas

managed by the original file system enables such changes without changing disk-area structure of the original file system.

Please amend the paragraph bridging pages 34 and 35, from line 19 on page 34 to line 16 on page 35, as follows

An object of an embodiment 3 is to share a disk without using the host table 110 used in the embodiment 1. Fig. 19 is a diagram illustrating a data sharing method of the embodiment 3. The host 1 (151) and the host 2 (152) specify different disk IDs (1903, 1907) for the disk array controller 122 through the SAN 153, and transfer commands (1902, [[1907]] <u>1906</u>) and data (1901, 1905). In Fig. 19, the following example is shown: the host 1 specifies a disk ID = 3; and the host 2 specifies a disk ID = 4. The disk array controller 122 is characterized by the following: if the disk array controller 122 is accessed with a specified disk ID = 3, the disk array controller 122 performs a usual disk input/output; and if the disk array controller 122 is accessed with a specified disk ID = 4, the disk array controller 122 performs input/output processing by utilizing the partition information after conversion 1301, the file management information after conversion 1401, 1402 shown in Fig. 15 of the embodiment 1. More specifically, this means that the disk ID = 3 is used only for the OS 1 (default file system) operating in the host 1, and that the disk ID = 4 is used only for the OS 2 operating in the host 2. A control method for each kind of OS is described in a disk control table 1908.

Please amend the paragraph on page 38, from lines 5 through 19, as follows:

Fig. 24 is a diagram illustrating a configuration of a whole system that uses a storage system of the embodiment 5 according to the present invention. A large part of the configuration shown in Fig. 24 is the same as that shown in Fig. 1. However, the configuration in Fig. 24 is different from that shown in Fig. 1 on the following points: a lock table 2401 and a lock control program 2402 are provided in the disk array controller 122; and a disk-array write program [[2403]] is changed. The lock control program 2402 is a program for controlling write operation so that discrepancy in data does not occur when a write request is issued from a certain host to a file that is shared among a plurality of hosts. In the lock table 2401, information, such as whether or not there is a file lock, is stored.

Please amend the paragraphs on pages 39 through 44, from line 24 on page 39 through line 12 on page 44, as follows:

Fig. 27 illustrates a flowchart of the lock control program [[2404]] 2402. In a step 2701, whether a request from the host is a command for a lock or a command for an unlock is judged. If the request is the command for a lock, LOCK processing ~~of a step 3002~~ is executed (step 2702). If not, UNLOCK processing is executed (step 2703).

Fig. 28 illustrates a flowchart of LOCK processing. In a step 2801, whether or not the file, for which the host has issued a write request, is in a locked state is judged. This can be judged by referring to the lock table 2401. For example, if a file as a target of the write request is the file A, the file is now in a locked state (refer to Fig. 25). If the file is in the locked state, the process proceeds to a step 2802. If not, the process proceeds to a step 2804. In a step 2802, whether or not the [[own]] owner host has locked the file is judged. If the [[own]] owner host has locked the file, the process proceeds to a step 2803. If not, the process proceeds to a step 2808 where the host is notified of failure of locking. In a step 2803, whether or not the same lock owner has locked the file is judged. If the same owner has locked the file, the process proceeds to the step 2804. If not, the process proceeds to the step 2808 where the host is notified of failure of locking. In the step 2804, a host ID is registered in a corresponding column in the lock table 2401. In a step 2805, an owner is registered in a corresponding column of the lock table 2401. In a step 2806, one is added to the lock count in the lock table 2401. For example, when locking the file B, a lock count of the file B is updated to "1". In a step 2807, the host is notified of success of locking.

Fig. 29 illustrates a flowchart of UNLOCK processing. In a step 2901, one is subtracted from a lock count, which is registered in the lock table 1401. For example, when unlocking a file C, a lock count of the file C is updated to "1". In a step 2902, whether or not the lock count is 0 or less is judged. If the lock count is 0 or less, the process proceeds to a step 2903. If not, the process proceeds to end ~~a step 2906~~. If the lock count is 0 or less, this means that all locks are unlocked. Therefore, in the step 2903, a host ID in the lock table 2401 is deleted. In a step 2904, an owner in the lock table [[2754]] is deleted. In a step 2905, a lock count in the lock table [[2754]] is set to 0. In the step 2906, the host is notified of success of UNLOCK.

Fig. 30 illustrates a flowchart of the disk-array write program [[2403]] of the embodiment 3. In a step 3001, inspection is performed to check whether or not it is locked. A flowchart of the lock check is shown in Fig. 31.

In Fig. 31, in a step 3101, a file name is searched from a disk address of a disk, which has been written, and from file management information. In a step 3102, with reference to the file name and the lock table 2401, whether or not the file is locked is judged. If the file is

locked, the process proceeds to a step 3103. If not, the process proceeds to a step 3106. In the step 3103, whether or not the host, which has locked the file, is the same as the [[own]] owner host is judged. If the host is the same as the [[own]] owner host, the process proceeds to a step 3104. If not, the process proceeds to a step 3105. In the step 3105, after waiting for given time, the step 3102 is executed again. In the step 3104, whether or not it is the same owner is judged. If it is the same owner, the process proceeds to the step 3106 where use is allowed. If not, the process proceeds to the step 3105. As a result, the inspection of locking is completed.

Fig. [[31]] 30 will be described again. If permission for use is gotten, the process proceeds to the step 3002 where a disk ID, which has been transferred by the host, is obtained. The disk ID is an identifier that is transferred from the host to the disk array controller together with an input/output request when, for example, the host issues the input/output request to the disk array controller. In a step 3003, a host table [[2401]] is obtained. In a step 3004, whether or not it is an original file system of the host, which has issued an access request, is judged. If it is the original file system, the process proceeds to a step [[3311]] 3011. If not, the process proceeds to a step 3005. The original file system means the one which has stored the file into the disk device. This embodiment is based on the assumption that the original file system is the file system of the OS 1. Steps 3005 through 3010 are executed when there is a request from a file system, which is different from the file system that has stored the file in the disk device. To be more specific, when an access request is issued from the file system of the OS 2, the steps are executed. In the step 3005, whether or not it is an access to the OS 2 boot program is judged. If it is the access to the OS 2 boot program, the process proceeds to the step 3008. If not, the process proceeds to the step 3006. In the step 3008, data, which has been transferred from the host, is written into the OS 2 boot program. In the step 3006, whether or not it is an access to partition information is judged. If it is the access to the partition program, the process proceeds to a step 3009. If not, the process proceeds to a step 3007. In the step 3009, data, which has been transferred from the host, is written into partition information after conversion. In the step 3007, whether or not it is an access to file management information is judged. If it is the access to the file management information, the process proceeds to the step 3010. If not, the process proceeds to a step 3011. In the step 3010, data, which has been transferred from the host, is written into file management information after conversion. The meaning ~~Meaning~~ of the boot program, the partition information after conversion [[,]] and the file management information after conversion are [[is]] similar to that described in the embodiment 1. In the step 3011,

usual disk-array write processing is performed, and data is written into a requested disk address. In a step 3012, a result of a write operation is transferred to the host. As a result of processing described above, even if a plurality of hosts write simultaneously, serialized processing can avoid discrepancy in data caused by a simultaneous write.

Please amend the paragraphs on pages 47 through 48, from line 5 on page 47 through line 25 on page 48, as follows:

Fig. 37 is a diagram illustrating the split function of a disk array. The disk array controller 122 has a (mirroring) function of writing the same data in a disk 1 (3705) and a disk 2 (3706) in order to improve input/output performance and reliability. The split function comprises split control and merge control. Split control 3702 permits a host to access two mirrored disks as two independent disks temporarily. Merge control 3703 changes the disks, which have been divided into two, back into one at the time of completion of processing by the host. In addition to it, the merge control 3703 reflects data, which has been updated while the disk was divided into two, in each disk. In this embodiment, in addition to these functions, a mirrored-disk conversion control program [[4204]] 3704 is provided to apply the split function to file sharing. As a result of the mirroring, contents of two disk devices 3705, 3706 become the same. Data structure of each individual disk is the same as that shown in Fig. 17.

Fig. 38 illustrates a flowchart of the mirrored-disk conversion control program 3704. As regards steps similar to those described in Fig. 12, description will be omitted. This flow is characterized in that synchronization control of a step 3801, split control of a step 3802, and conversion result of a step 3803 are reflected in the disk 2. In the step 3802, the split control program 3702 is started, and the disk is divided into the mirrored disk 1 (3705) and the mirrored disk 2 (3706). In a step 3803, converted file information is stored in either of the divided disks. More specifically, as described in the embodiment 1, the OS 2 boot program 1501, the partition information after conversion 1301, and the file system information 1401, 1402 after conversion are stored on either of the disks. As a result of the processing described above, one of the mirrored disks becomes a disk that stores information about the file system, which is different from the original file system. Fig. 39 shows a state of a disk after the split control of this embodiment is executed. Using the disk 1 ([[3701]] 3705) as an OS 1's dedicated disk for input/output, and using the disk 2 ([[3702]] 3706) as an OS 2's dedicated disk for input/output, result in distributed accesses to different disks even when sharing a disk from a plurality of hosts. Therefore, it is possible to avoid decrease in

performance caused by conflict. When data in one disk is updated, the updated contents are reflected by the step 3803.